Compte Rendu TD08

Frames *

1. Construire la représentation

- Donnez un algorithme décrivant ce que doit faire la fonction make-individu dont voici un appel :

(make-individu 'ELEPHANT '(Name "Clyde" COLOR "grey" AGE 5))

- Écrivez le code Lisp de cette fonction.

Note: on utiliser les données initiales suivantes:

```
(setq ELEPHANT
      '(ELEPHANT
        (TYPE (VALUE . CONCEPT))
        (IS-A (VALUE . ETRE))
        (COLOR (DEFAULT . GREY))
        (AGE (IF-NEEDED ask-user)
             (IF-ADDED check-age))
        (POIDS (IF-NEEDED computer-weight-from-age))
        (AFFICHAGE (IF-ADDED draw-elephant)
                   (IF-REMOVED erase-elephant))))

(pushnew 'ELEPHANT *frames*)
```

```
;;1. si le frame concept n'exist pas erreurs
;;2. creer un id unique
;;3. creer un début de frame avec le nom et individu
;;4. pour chaque slot .vérifier que le slot est autorisé  .si oui i faut créer slot
avec la valeur correspondante est ajouter au début de frame précédent
;;5. ajouter id à *frame*
;;6. retourne id
```

```
(defun make-individu(name concept prop_val)
    (if (not (member concept *frames*)) ; si le concept n'est pas présent
        (format t "~& ~A non initialisé" concept) ; on renvoit un message
d'avertissement
        (let ( (new nil) (N (gentemp "N")) ) ; sinon on va effectivement
créer l'individu
            (push (list 'type (cons 'value 'individu)) new)
            (push (list 'is-a (cons 'value concept)) new) ; on ajoute
tout d'abord les slot IS-A et TYPE
            (loop
                (let ( (property (car prop_val)) (value (cadr
prop_val)) )
                    (cond
```

```
                        ( (numberp value)  (push (list 'age   (cons 'value
value)) new) )
                        ( (Stringp value)  (push (list 'color (cons 'value
value)) new) )
                    )
                )
                (setq prop_val (cddr prop_val))
                (when (equal prop_val nil)  (return new)  )
            )
            (set N new)
            (pushnew N *frames*)
        )
    )
)
```

- Test et Résultat :

```
(setq val '(color "blue" age 5 ) )
(make-individu 'LIU 'ELEPHANT val)
(print *frames*)
(print N1)
```

```
YanLIUdeMacBook-Pro:Lisp yann$ clisp TD08.lisp
(N1 ELEPHANT)
((AGE (VALUE . 5)) (COLOR (VALUE . "blue")) (IS-A (VALUE . ELEPHANT))
 (TYPE (VALUE . INDIVIDU)))
YanLIUdeMacBook-Pro:Lisp yann$
```

- Fonction get-slot-value

```
(defun get-slot-value (frame slot)
  ; on cherche la valeur du slot dans le frame
  ; frame : nom du frame
  ; slot : nom du slot
   (if (not (member frame *frames*))
      (format t "~& ~a is not present as a frame" frame) ; on indique si le frame n'existe pas
      (let ((props (symbol-value frame)) val) ; on récupère toutes les slots
         (setq val  (cdr(cadr (assoc slot props))))
         val
      )
   )
)
```

- Test et Résultat :

```
(print (get-slot-value 'N1 'AGE) )
(print (get-slot-value 'N1 'COLOR) )
```

```
(N1 ELEPHANT) ; *frames*
((AGE (VALUE . 5)) (COLOR (VALUE . "blue")) (IS-A (VALUE . ELEPHANT))
 (TYPE (VALUE . INDIVIDU)))
5
"blue"
YanLIUdeMacBook-Pro:Lisp yann$
```